## Problem Understanding :

After exploring the data we can observe that the objective of the recommendation engine is to look at past user-poll interaction data and gauge which polls would the users be more likely to interact with in the future.

The past engagements with the user and polls have been - impressions, expansion, sharing, answering polls. By using these we need to understand which polls will the user definitely engage with next.

"Session" tells us when someone starts or stops using the app, like logging in or logging out. "Impression" is when someone just looks at a poll but doesn't join in. If someone expands a post to see the comments, that's called an "Expand" event. "Shares" is when someone shares a post, and "Polls Answered" is when someone takes part in a poll by voting for an option. So, the table helps us understand what users are doing in the app—whether they're logging in, just looking at things, sharing posts, or participating in polls.

## Data Preprocessing (/Handling) :

We will investigate the data column by column, for each type of data. The information can be broken down into :

* Event Specific Data
* User Properties
* Poll characteristics

For each category, an column by column investigation has been detailed

### 1. Events Specific Data :

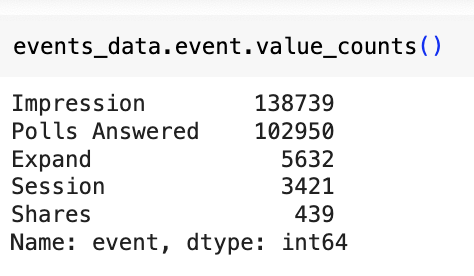
#### user\_code & poll\_code

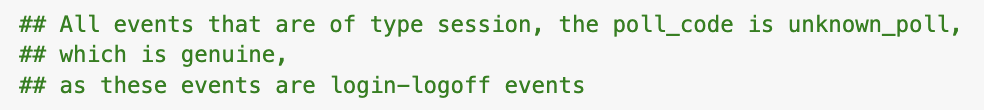
* We can see there are unique 4499 users’s event data
* And we have unique 893 poll’s interaction data



* We also observe that the maximum user\_id is 4835, which implies that some users don’t have any events, ie, they have not used the app
* We also observe that the maximum poll\_code is 920, which implies that some polls don’t have any events, ie, they have not been engaged by any user

#### event

* Based on the events data the following is the break-up of each events counts   
  
* From this we can observe that the event “session” is much lower than the total users. Event session’s occur in pairs, and thus the total data available at max can be for 1700 users. Based on this a call was taken to ignore the session data for the modeling purpose.
  + Users engagement is session’s can be explored from an analytics & insight point-of-view, but the total counts are flow to find similar behavior users.



* Before *answering a poll*, you must have an Impression Event, as it can be understood from the description of the event. Same for *Shares* and *Expand.* Below table shows the percentage of expected events that were missing, and thus, those events were created and appended. The timestamp of the proceeding event is assumed as the timestamp of the impression event. For example a user shared a poll at a timestamp of 2022/12/5 12:22 and the *Impression,* for this poll was missing, a new impression was created with the same timestamp for that user-poll air.

| **Event Type** | **Event Count** | **Had Impressions** | **Didnt have Impressions** | **Percentage Missing** |
| --- | --- | --- | --- | --- |
| *Shares* | 342 | 322 | 20 | 5.85% |
| *Expands* | 4642 | 4476 | 166 | 3.58% |
| *Poll Answered* | 102916 | 83751 | 19165 | 18.62% |

#### id\_code

* id\_code is not clear. Unexpectedly even Session have an option\_id.
* It makes sense that there is id\_code for the Polls Answered event, but having an id\_code for any other event is unexpected.
* Thus, it is different from the description.
* we shall ignore the column

### 2. User Feature

On a total 4499 users’ properties were provided. 5% of total user base would be 225, and this if any particular bucket was less than 5% it was ignored.

* **Country -** Out of 4499 users, 4285 users belonged to the same country
* **City\_code -** City\_1 and City\_18 had 3351 and 238 users respectively. All other cities had very few users.
* **Gender** - Males and Females were 1668 and 310 respectively. *Skipped the question* and *nulls* are considered differently as this would be generated due to changes in onboarding flow. Which would signify the age of the user in the platform. Thus we retained *Skipped* and *null* separately.
* **College** - Either College Code was present, or college code was not present, in the drop down.

### 3. Poll Features

### 

* While there are 920 unique polls in the events data, the features that is available for 747 unique polls
* 60 polls have duplicate record of category Random
* Those records were removed

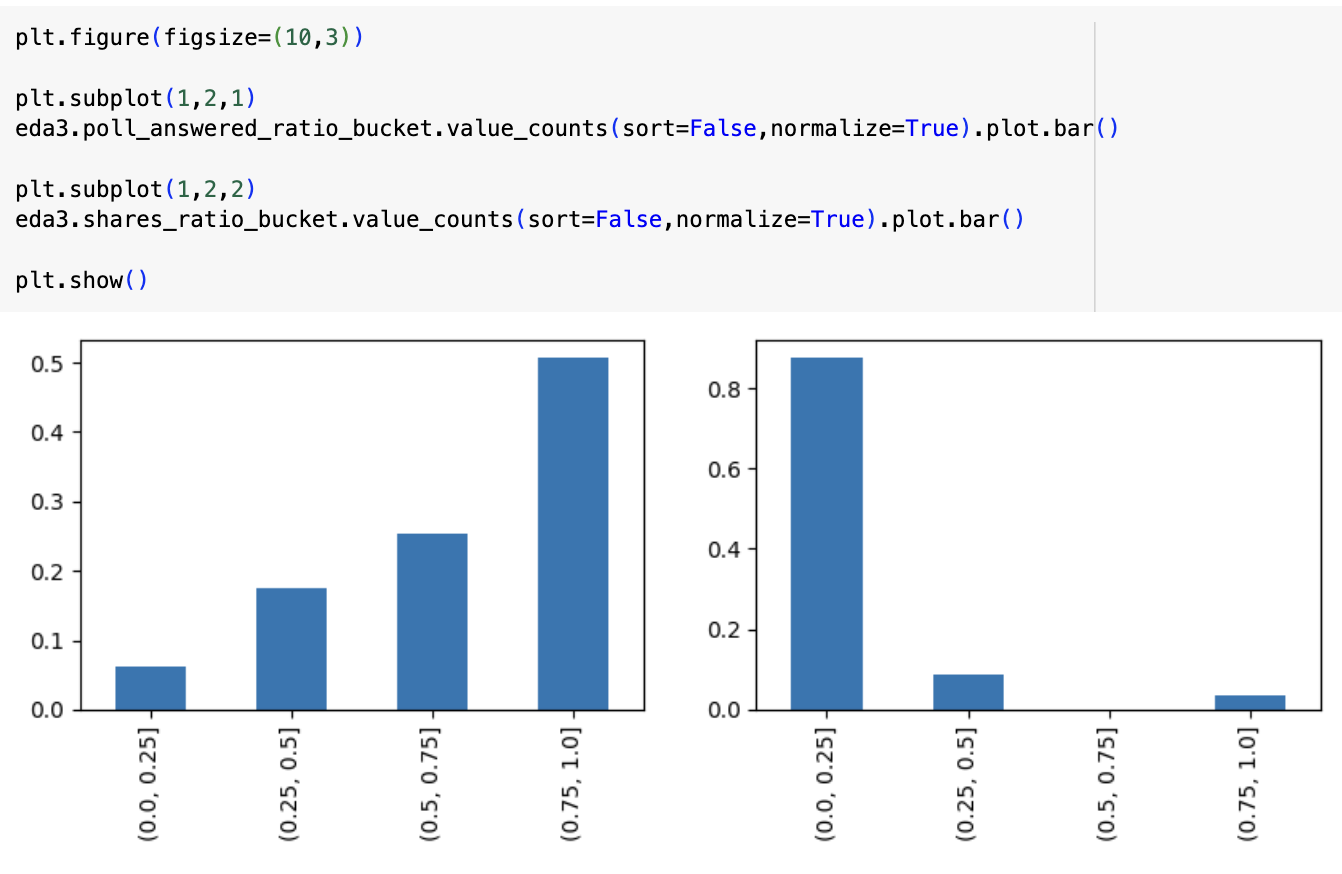
### 4. App Events Data - pre-processing for target generation

* *Session* events were removed
* *Unknown\_user* event were removed
* Events data were rolled up at user-poll interaction level
* Each type of event was given a score
* Finally Impression and Poll answered were prioritized as targeted goals
* This can also be looked at as prioritizing engagement on app

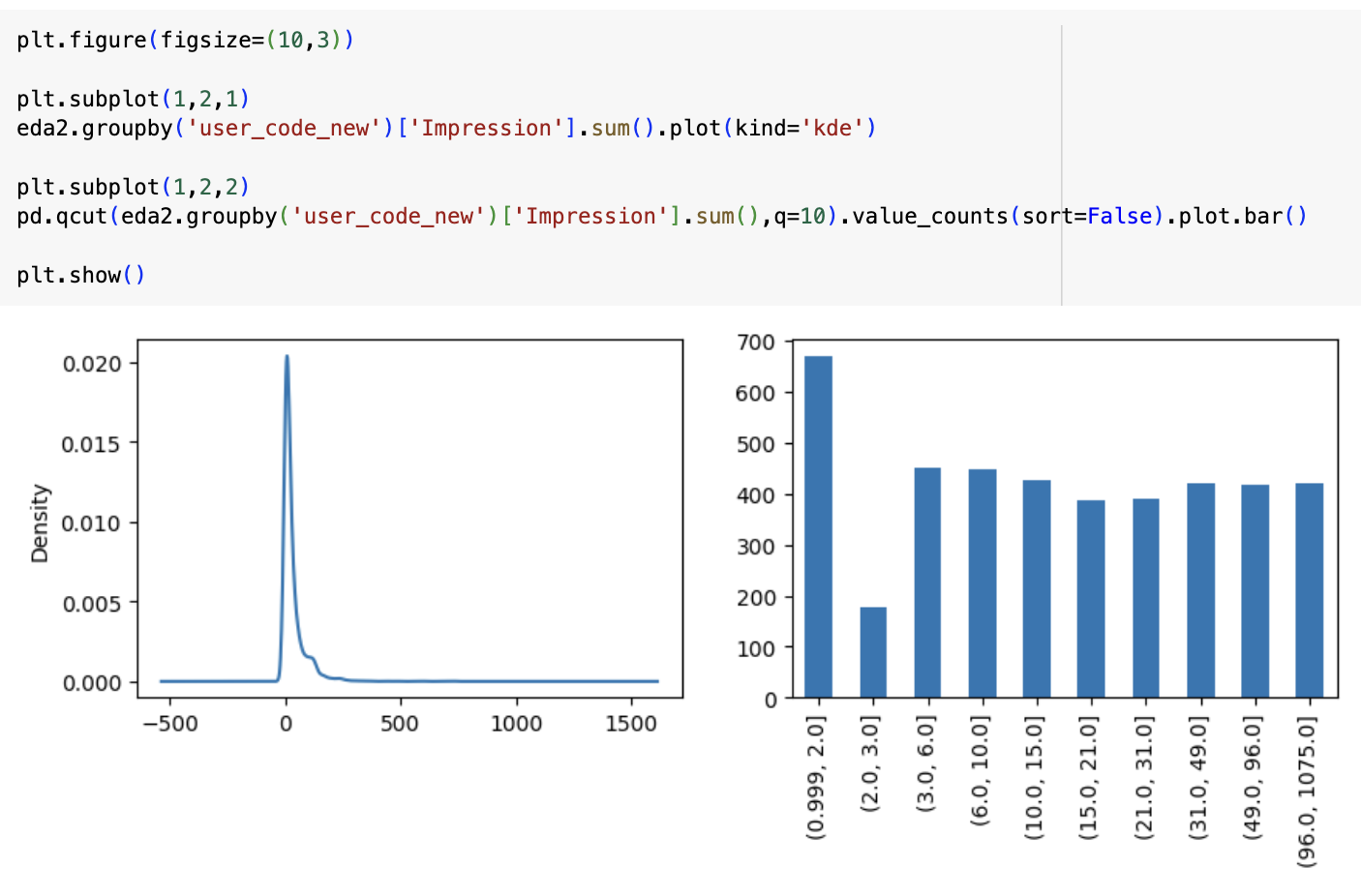


## Exploratory Data Analysis

* More than 50% users answer more than 75% polls they read
* 90% users share < 25% of the polls they read

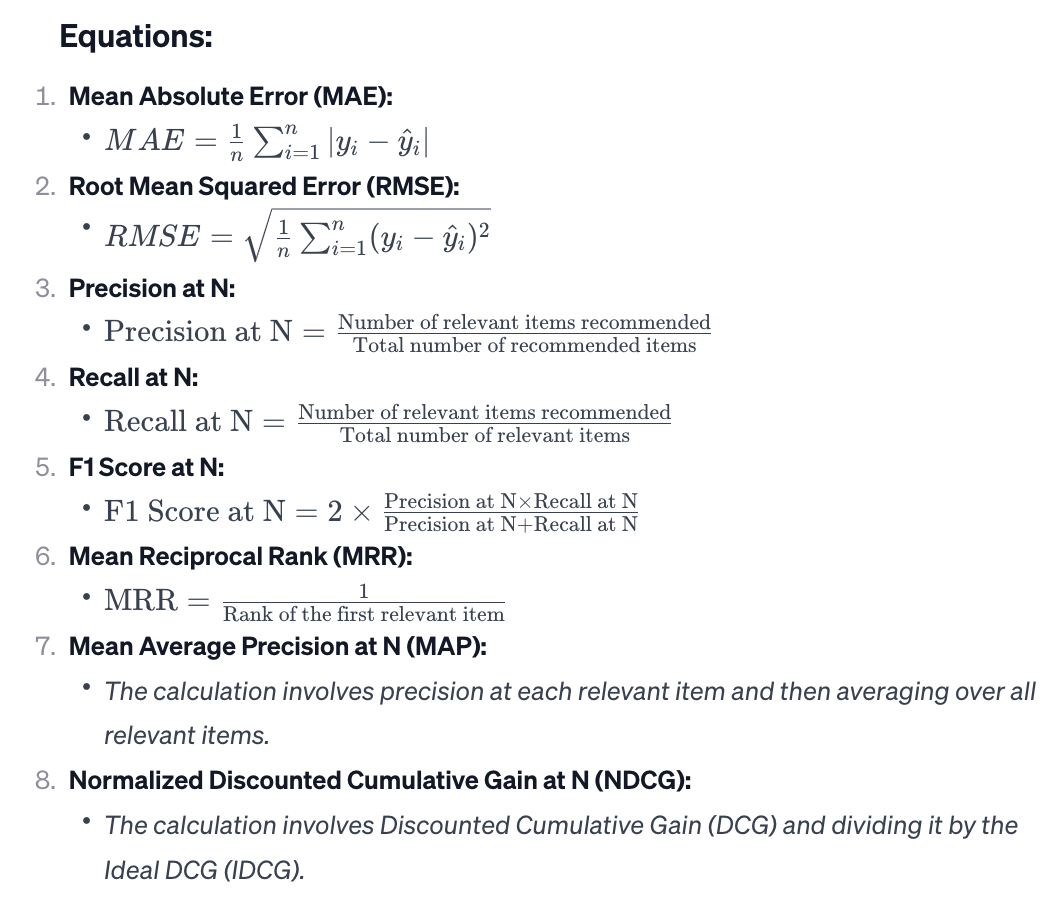


* A large set of users drop off before reading 3 polls. After that drop off is consistent.



## Metrics Considered

* Mean Absolute Error (MAE):
  + Definition: The average absolute differences between the predicted and actual values.
  + Explanation: MAE measures the average magnitude of errors, without considering their direction.
* Root Mean Squared Error (RMSE):
  + Definition: The square root of the average of squared differences between predicted and actual values.
  + Explanation: RMSE gives more weight to large errors, making it sensitive to outliers.
* Precision at N:
  + Definition: The proportion of recommended items that are relevant out of the total recommended items.
  + Explanation: Precision measures the accuracy of the recommendations by assessing the ratio of relevant items in the top-N list.
* Recall at N:
  + Definition: The proportion of relevant items that are successfully recommended out of all the relevant items.
  + Explanation: Recall assesses the ability of the system to capture all relevant items in the recommendations.
* F1 Score at N:
  + Definition: The harmonic mean of precision and recall, providing a balance between the two.
  + Explanation: F1 Score considers both precision and recall, making it suitable for imbalanced datasets.
* Mean Reciprocal Rank (MRR):
  + Definition: The average of the reciprocal ranks of the first relevant item in the recommendation list for each user.
  + Explanation: MRR focuses on the quality of the top recommendation for each user.
* Mean Average Precision at N (MAP):
  + Definition: The average precision calculated for each user and then averaged over all users.
  + Explanation: MAP emphasizes precision at each relevant item and averages over users to evaluate the overall recommendation quality.
* Normalized Discounted Cumulative Gain at N (NDCG):
  + Definition: A measure of the ranking quality that considers the position of relevant items in the recommendation list.
  + Explanation: NDCG gives higher importance to relevant items that appear higher in the recommendation list.





## Methods

While most of the methods are explained along the document, here I have highlighted some specific methods that were missing in the documentation

### Sampling Method

Stratified Sampling was performed

## 

### Scoring Method

Weighted Rating is used to calculate the top performing polls in each region

## 

## Algorithms Tried

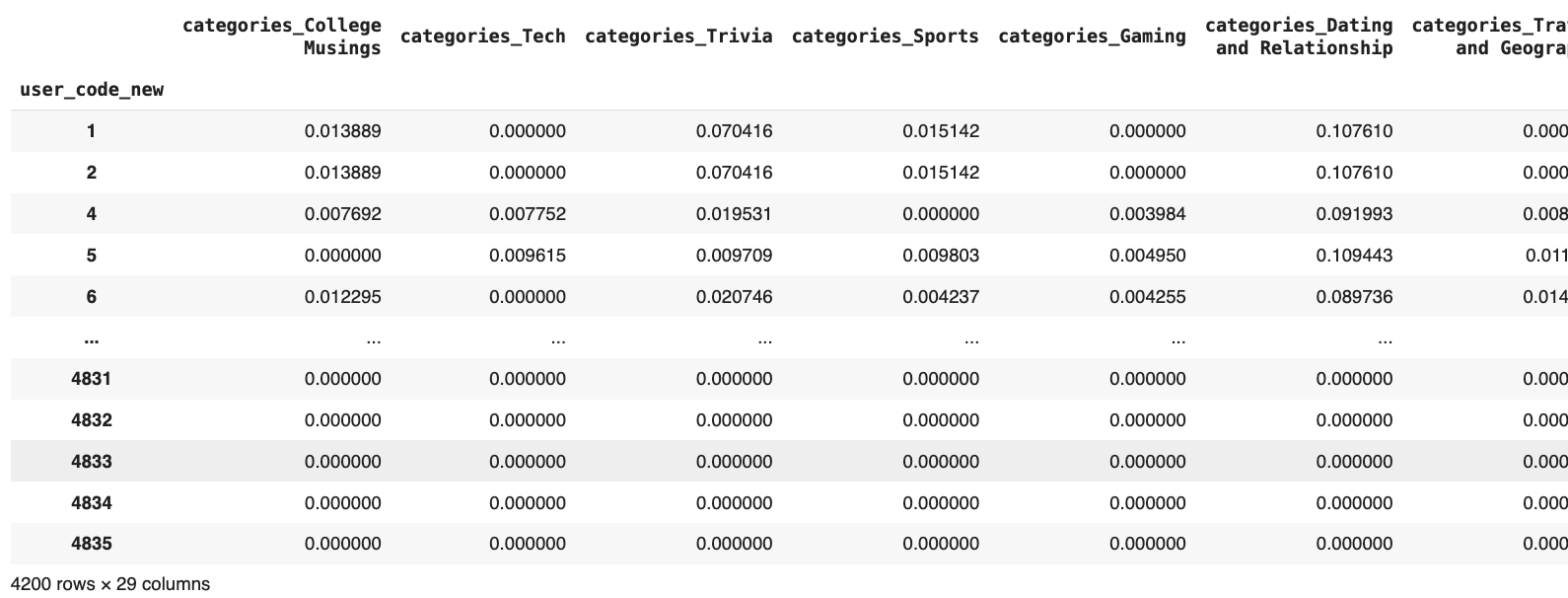
This section will elaborate on the following algorithms being used. It will also share the performance of each of these algorithms and suggest the best performing algorithm, with fair justification.

* Content Based Algorithm
* Collaborative Learning - User Based
* Collaborative Learning - Item Based
* Singular Value Decomposition (SVD) - User Based Collaborative Filtering
* Collaborative Filtering + Cold Start Solution by Appending Demographic Data before estimating user-user similarity
* Collaborative Filtering + Cold Start Solution by Appending Demographic Data before estimating user-user similarity
* Demographic Basis Trending
* Hybrid approach
  + Singular Value Decomposition (SVD) - User Based Collaborative Filtering
  + Content Based Algorithm
  + Demographic Basis - Most Trending

#### 1. Content-Based Algorithm:

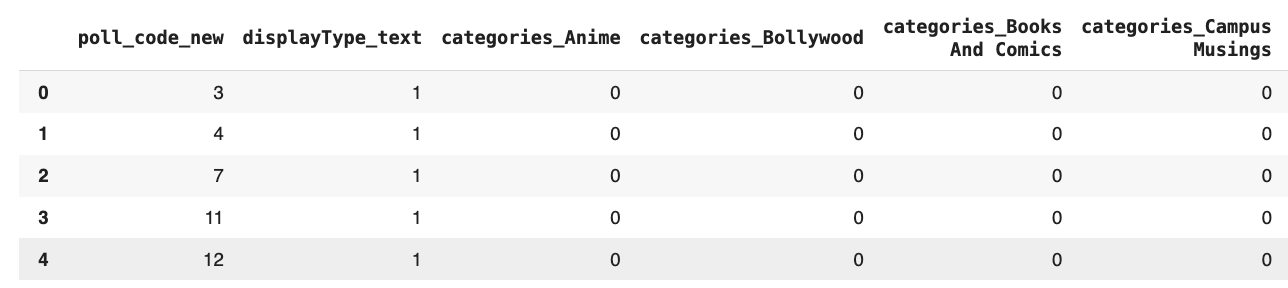
Content-based recommendation relies on the features of items and user preferences to make recommendations. It suggests items similar to those a user has liked in the past based on item features, such as keywords, genres, or content descriptions. For this a user characteristic matrix is created, which is of the size - user\_count x item\_characteristics. The second matrix of item\_count x item\_chracteristics. The multiplication of these will give the score for each item the user has not engaged with, based on his past experience.

The following image is the users preference matrix :



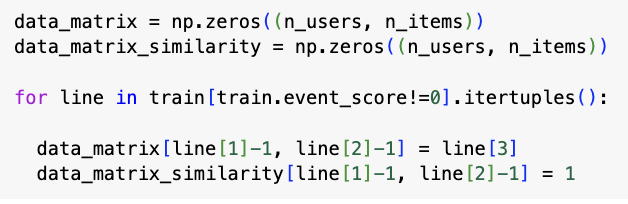
### 

The following is the poll characteristic matrix :

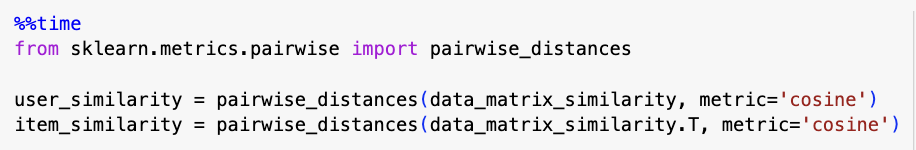


#### 2. Collaborative Learning - User Based:

User-based collaborative filtering recommends items to a user based on the preferences and behavior of users with similar tastes. It identifies users similar to the target user and recommends items they liked. The users-user similarity is based on the items they have already interacted with. And for estimating that the similarity, linear distance if two user’s user-items vector is compared. This distance can be a cosine distance or pearson correlation distance.



The following code checks for similarity between users :

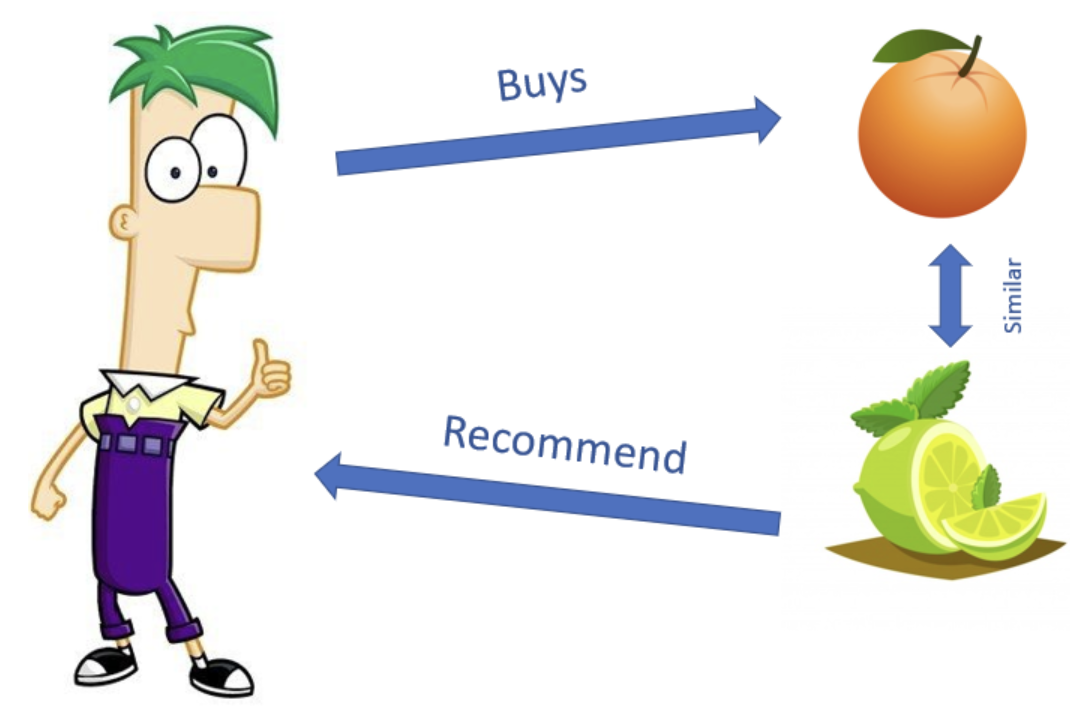


#### 3. Collaborative Learning - Item Based:

Item-based collaborative filtering recommends items similar to those a user has liked in the past. It identifies items similar to those the user has interacted with and recommends them.

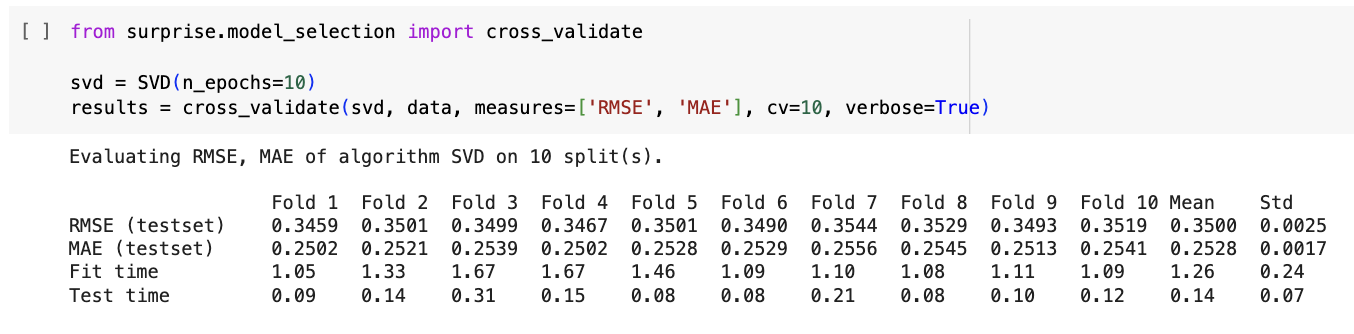
### 

The transposed form of the data\_matrix is used to find item-item similarity. The items that are similar to each other are then used to recommend items to the users.



#### 4. Singular Value Decomposition (SVD):

SVD is a matrix factorization technique that decomposes the user-item interaction matrix into three matrices. It represents users and items in a lower-dimensional space, capturing latent features. This approach is often used for collaborative filtering.



### 

#### 5. [Creative] Collaborative Filtering + Cold Start Solution by Appending Demographic Data before Estimating User-User Similarity:

This method addresses the cold start problem, where little information is available about new users. It appends demographic data (converted into categorical variables) to the user-item interaction matrix before estimating user-user similarity, enhancing the accuracy of recommendations for new users.

***💡This approach was creative as it avoided calculating user-user similarity based on demographics separately. This solution can reduce computation cost, to solve cold start problem***

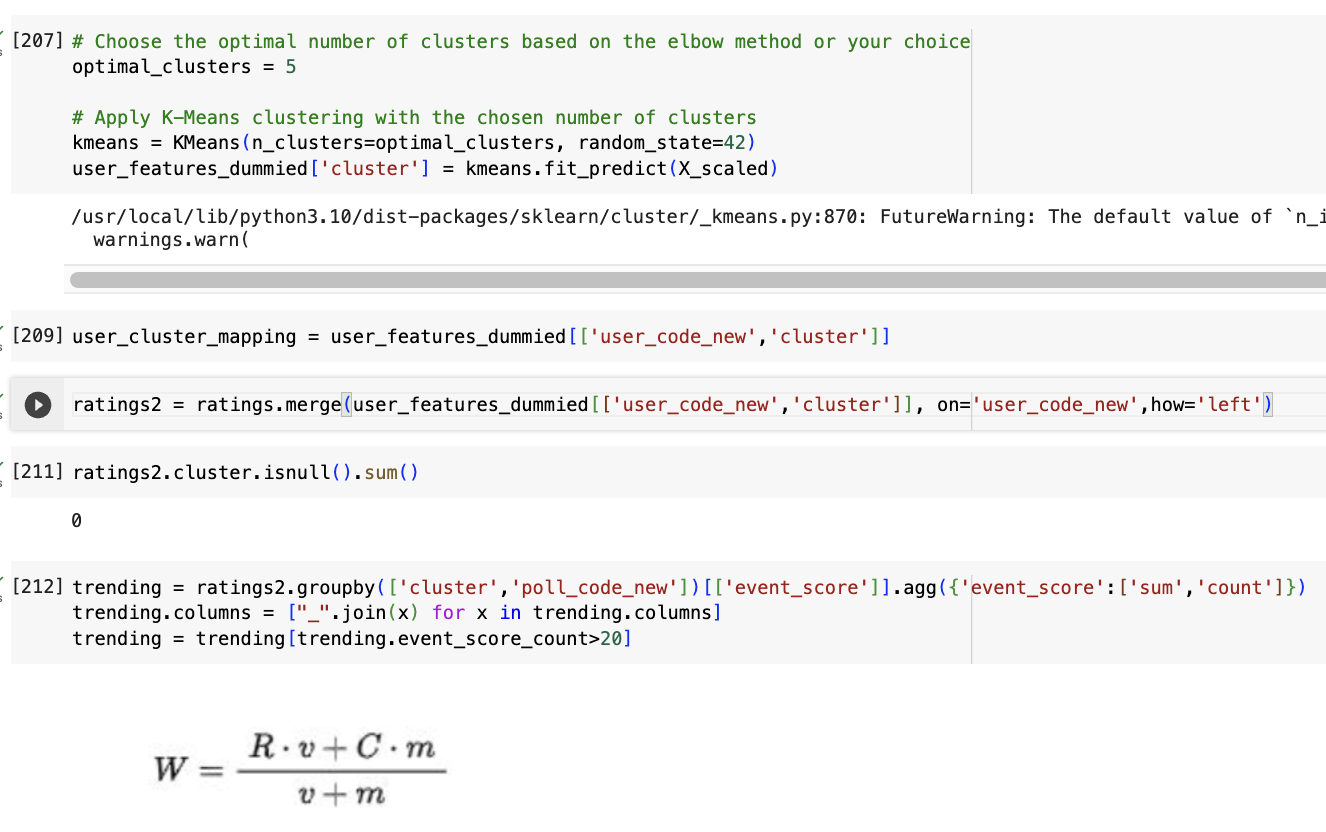
### 



#### 6. Demographic Basis Trending:

Demographic basis trending recommends items based on current trends and user demographics. It considers factors like age, gender, or location to tailor recommendations to specific user segments. The demographic data is used to find the top performing post, in each of the demographics. Users in those regions were recommended the posts that were performing well in those regions.

Weighted Rating was used to calculate the significant polls. The formula for which is shared in the image below.

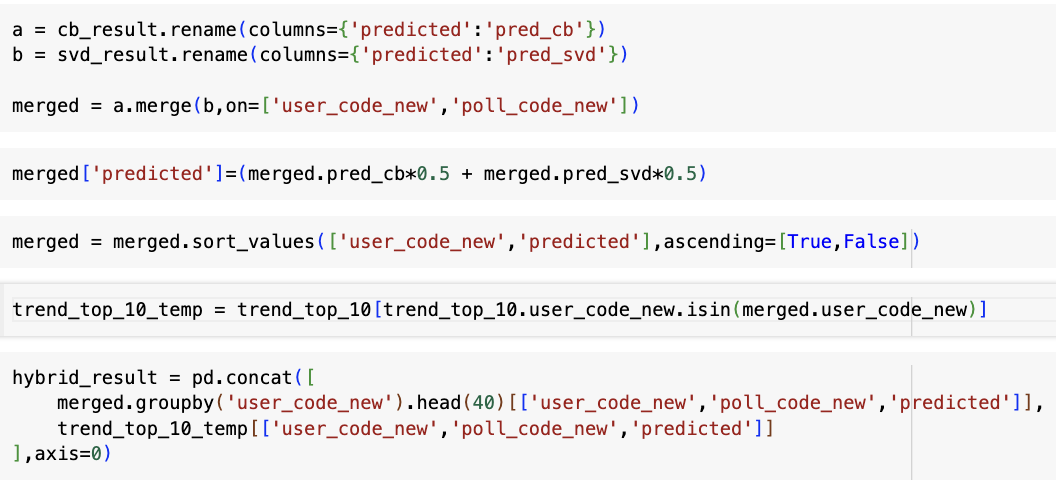


### 

#### 7. [Creative] Hybrid Approach: Singular Value Decomposition (SVD) - User Based Collaborative Filtering + Content-Based Algorithm + Demographic Basis - Most Trending:

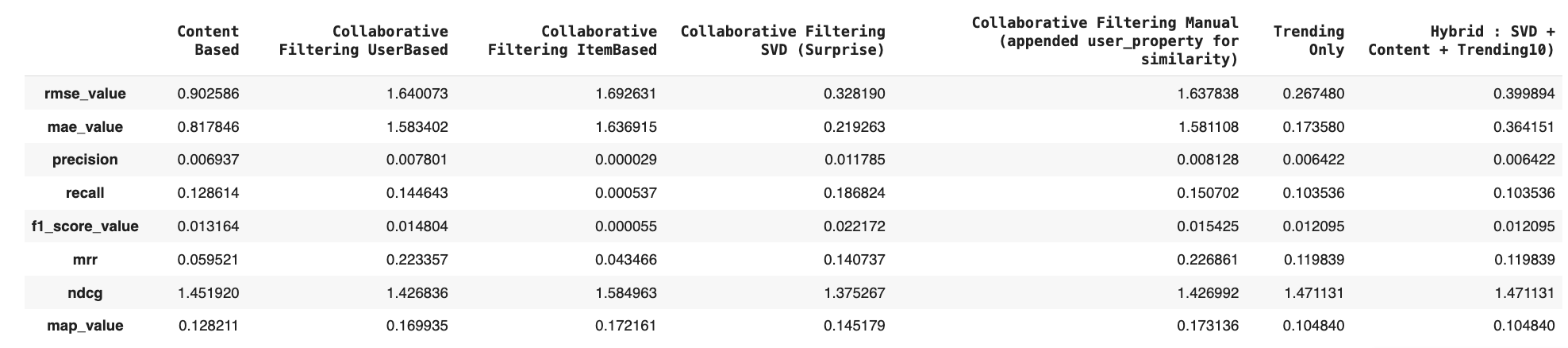
Explanation: This hybrid approach combines multiple recommendation strategies. The top 10 recommendations from each region was appended with the top 40 recommendations based on other approaches. It uses SVD, a user-based collaborative filtering to capture latent features and user preferences. It also incorporates content-based algorithms to consider item characteristics. Both these algorithms are given equal weightage.

***💡This approach was creative as it first focused on what are the top trending polls in the same demographic users. Since those posts have higher potential to be viral.***



### Best Performing Algorithm

**Based on NDCG SVD performs the best**



### Why NDCG?

Normalized Discounted Cumulative Gain (NDCG) is like a smart way of figuring out how good a recommendation system is. When we compare it to other methods like MAP, Precision at K, and Recall at K, NDCG is considered better because it looks at not just whether the recommended things are right (like Precision and Recall do) but also where they are in the list. Precision at K and Recall at K only care about having the right stuff in the recommendations, not where they are. MAP looks at each right thing, but it might not care enough about where it is in the list. NDCG is clever because it considers both things – it looks at how many right things are in the list and also cares about where they are. It gives more importance to things at the top of the list. This makes NDCG a better way to measure how well a recommendation system is doing, especially when the order of the recommendations matters a lot to users.

| **Metric** | **NDCG** | **Precision@k** | **Recall@k** | **Mean Avg Precision (MAP)** |
| --- | --- | --- | --- | --- |
| *Accounts*  *for Position* | Yes | Partially (up to k) | Partially (up to k) | No (only order) |
| *Accounts*  *for Relevance* | Yes | Yes | Yes | Yes |
| *Accommodates*  *Graded Relevance* | Yes | No (Binary) | No (Binary) | Sometimes |
| *Holistic Measure* | Yes | No | No | No |

## 

## 

## 

## 

## 

## 

## 

## 

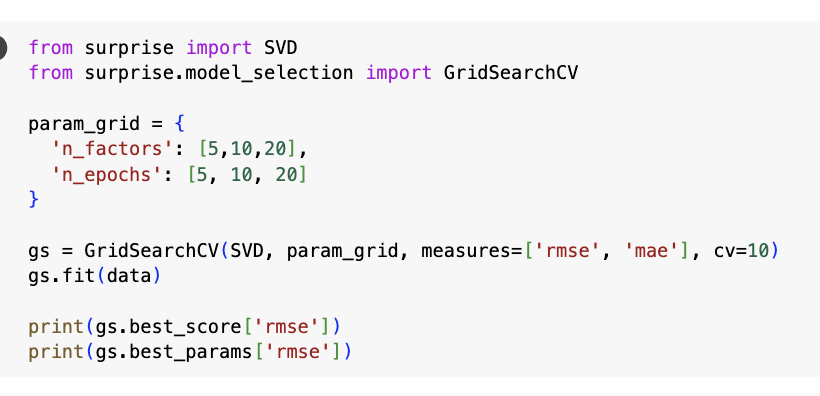
## 

## 

## 

## Hyperparameter Tuning

| **Hyperparameter** | **Description** | **Default Value** |
| --- | --- | --- |
| **n\_factors** | **Number of factors (latent features)** | **100** |
| **n\_epochs** | **Number of iterations for SGD optimization** | **20** |
| init\_mean | Mean of normal distribution for factor vectors init. | 0 |
| init\_std\_dev | Std. dev. of normal distribution for factor vectors init. | 0.1 |
| lr\_all | Learning rate for SGD optimization of all parameters | 0.005 |
| reg\_all | Regularization term for all parameters | 0.02 |





## Future Work

1. SVD vs User Based Collaborative Learning :

They both perform differently - why? Theoretically SVD is a matrix factorization step to reduce computational cost.

1. Temporal Understanding has not been accounted
2. OOT sample is not taken for recommendation
3. Session level estimation of users engagement type has not been considered